#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include "base64.h"

// -----------------------------------------------------------------------------

// Your network SSID and password

const char\* ssid = "RowanWiFi";

// const char\* password = "YourNetworkPassword";

const char\* account\_sid = "ACd574289e1a602ded89fcb039807fbf9b";

const char\* auth\_token = "7eee4d19f0c2e4718f07545d6b90ab04";

String from\_number = "+18775914527";

String to\_number = "+16097749264";

String message\_body = "There is a Fire, Send Help.";

// Find the api.twilio.com SHA1 fingerprint using,

// echo | openssl s\_client -connect api.twilio.com:443 | openssl x509 -fingerprint

const char fingerprint[] = "93:B5:96:8C:BC:63:8D:A1:D5:AD:1E:DA:30:A1:DB:04:73:DC:BC:24";

char i;

// -----------------------------------------------------------------------------

// Switch and LED light pins

#define PIN\_PULL\_UP1 5 // NodeMCU label: D1, GPIO pin# 5

#define LED\_ONBOARD\_PIN 2 // NodeMCU label: D4, GPIO pin# 2

#define LED\_PIN LED\_ONBOARD\_PIN

// -----------------------------------------------------------------------------

String urlencode(String str) {

String encodedString = "";

char c;

char code0;

char code1;

char code2;

for (int i = 0; i < str.length(); i++) {

c = str.charAt(i);

if (c == ' ') {

encodedString += '+';

} else if (isalnum(c)) {

encodedString += c;

} else {

code1 = (c & 0xf) + '0';

if ((c & 0xf) > 9) {

code1 = (c & 0xf) - 10 + 'A';

}

c = (c >> 4) & 0xf;

code0 = c + '0';

if (c > 9) {

code0 = c - 10 + 'A';

}

code2 = '\0';

encodedString += '%';

encodedString += code0;

encodedString += code1;

}

yield();

}

return encodedString;

}

// -----------------------------------------------------------------------------

String get\_auth\_header(const String& user, const String& password) {

size\_t toencodeLen = user.length() + password.length() + 2;

char toencode[toencodeLen];

memset(toencode, 0, toencodeLen);

snprintf(toencode, toencodeLen, "%s:%s", user.c\_str(), password.c\_str());

String encoded = base64::encode((uint8\_t\*)toencode, toencodeLen - 1);

String encoded\_string = String(encoded);

// std::string::size\_type i = 0;

// Strip newlines (after every 72 characters in spec)

while (i < encoded\_string.length()) {

i = encoded\_string.indexOf('\n', i);

if (i == -1) {

break;

}

encoded\_string.remove(i, 1);

}

return "Authorization: Basic " + encoded\_string;

}

// -----------------------------------------------------------------------------

void connectWiFi() {

Serial.println("+ Connect to WiFi. ");

WiFi.begin(ssid, NULL);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println("");

Serial.print("+ Connected to WiFi, IP address: ");

Serial.println(WiFi.localIP());

}

WiFiClientSecure client;

void sendSms(int theValue) {

// Use WiFiClientSecure to create a TLS 1.2 connection.

// Note, using a cert fingerprint is required.

client.setFingerprint(fingerprint);

Serial.printf("+ Using fingerprint '%s'\n", fingerprint);

const char\* host = "api.twilio.com";

const int httpsPort = 443;

Serial.print("+ Connecting to ");

Serial.println(host);

if (!client.connect(host, httpsPort)) {

Serial.println("- Connection failed.");

char buf[200];

int err = client.getLastSSLError(buf, 199);

buf[199] = '\0';

Serial.print("- Error code: ");

Serial.print(err);

Serial.print(", ");

Serial.println(buf);

return; // Skips to loop();

}

Serial.println("+ Connected.");

Serial.println("+ Post an HTTP send SMS request.");

String post\_data = "To=" + urlencode(to\_number)

+ "&From=" + urlencode(from\_number)

+ "&Body=" + urlencode(message\_body) + String(theValue);

String auth\_header = get\_auth\_header(account\_sid, auth\_token);

String http\_request = "POST /2010-04-01/Accounts/" + String(account\_sid) + "/Messages HTTP/1.1\r\n"

+ auth\_header + "\r\n"

+ "Host: " + host + "\r\n"

+ "Cache-control: no-cache\r\n"

+ "User-Agent: ESP8266 Twilio Example\r\n"

+ "Content-Type: application/x-www-form-urlencoded\r\n"

+ "Content-Length: " + post\_data.length() + "\r\n"

+ "Connection: close\r\n"

+ "\r\n"

+ post\_data

+ "\r\n";

client.println(http\_request);

Serial.println("++ Message request sent.");

//

// Read the response.

// Comment out the following, if response is not required. Saves time waiting.

Serial.println("++ Waiting for response...");

String response = "";

while (client.connected()) {

String line = client.readStringUntil('\n');

response += (line);

response += ("\r\n");

}

Serial.println("+ Connection is closed.");

Serial.println("+ Response:");

Serial.println(response);

}

// -----------------------------------------------------------------------------

// Turn light on when the switch is flipped.

int counter = 0;

boolean setPullUpState1 = false;

void checkPullUpSwitch() {

if (digitalRead(PIN\_PULL\_UP1) == LOW) {

if (!setPullUpState1) {

Serial.println("+ Switch down.");

setPullUpState1 = false;

// Logic: switch flipped, circuit closed.

digitalWrite(LED\_PIN, LOW);

}

setPullUpState1 = true;

} else {

if (setPullUpState1) {

Serial.println("+ Switch up.");

setPullUpState1 = false;

// Logic: switch released.

counter++;

Serial.print("+ Loop counter = ");

Serial.println(counter);

sendSms(counter);

digitalWrite(LED\_PIN, HIGH);

}

}

}

// -----------------------------------------------------------------------------

void setup() {

Serial.begin(115200); // 115200 or 9600

delay(1000); // Give the serial connection time to start before the first print.

Serial.println(""); // Newline after garbage characters.

Serial.println(F("+++ Setup."));

pinMode(PIN\_PULL\_UP1, INPUT\_PULLUP);

pinMode(LED\_PIN, OUTPUT);

digitalWrite(LED\_PIN, LOW); // Off until WiFi connected.

connectWiFi();

digitalWrite(LED\_PIN, HIGH); // On

Serial.println(F("+ Starting the loop."));

}

// -----------------------------------------------------------------------------

void loop() {

checkPullUpSwitch();

delay (100);

}